Kom Ons Praat BASIC

Jac Nöthling

KOM ONS PRAAT BASIC

Jac Nöthling

ISBN 0 7959 0529 7

DAAN RETIEF UITGEWERS

Kom Ons Praat BASIC

Jac Nöthling

Illustrasies deur Dewald Raath



DAAN RETIEF UITGEWERS

Vir Melodie, Erika en Nini

Posbus 3570, Pretoria 0001

Omslagontwerp en illustrasies deur
Dewald Raath

Geset in Souvenir Medium en Colonial Bold

Fotoset en tipografie deur
Unifoto, Kaapstad

Gedruk en gebind deur
Printpak Boeke, Kaap

ISBN 0 7959 0529 7

KOM ONS PRAAT BASIC

Oor dit en dat

In hierdie boek gaan ons hoofsaaklik oor rekenaarprogrammering gesels, en in besonder oor *Basic*.
Die programmeerder is die "brein" agter die rekenaar. Hy of sy is soos die kaptein van 'n skip wat al die dinkwerk vir die skip en sy bemanning moet doen.
Programmering is eintlik niks nuuts nie - trouens, dit het al in 1832 begin en die wêreld se eerste programmeerder was 'n vrou! Dis net om te wys dat vrouens al in die vorige eeu hul plek in die samelewing kon vol staan!
Die rekenaar het 'n baie interessante geskiedenis. Diegene van julle wat

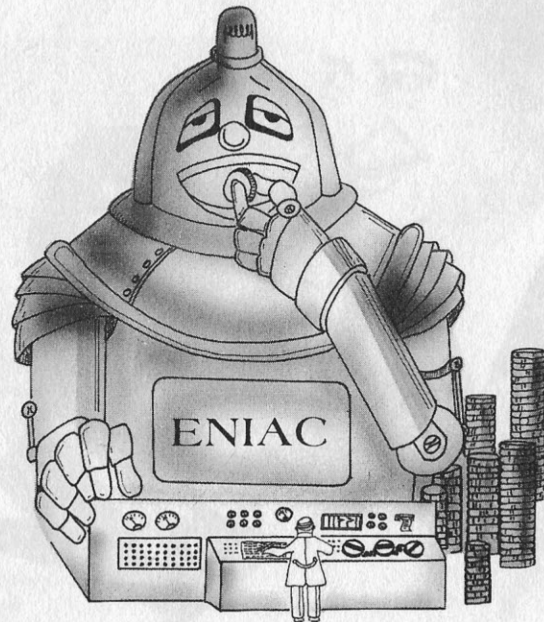


Twee persone wat die geskiedenis van die wêreld verander het. Links is dr. John von Neumann, uitvinder van die elektroniese rekenaar en regs is dr. Robert Oppenheimer, die man wat die eerste kernbom gebou het. Sonder 'n rekenaar sou dr. Oppenheimer en sy span wetenskaplikes nie die eerste atoombomme kon bou nie.

daarin belangstel kan gerus my boek *Rekenaars* (Daan Retief Uitgewers, 1985) lees. Daarin sal julle meer van die reuse *ENIAC* leer – 'n masjien met 'n massa van 30 ton. Terloops, *ENIAC* bestaan vandag nog en dit word in die Smithsonian Instituut se museum in Washington ten toon gestel.

Dr. John von Neumann word allerweë as die uitvinder van die wêreld se eerste volwaardige elektroniese rekenaar beskou. Sy goeie vriend, dr. Robert Oppenheimer, wat die eerste kernbomme gebou het, het baie nou met hom saamgewerk. Sonder 'n rekenaar sou dr. Oppenheimer en sy span wetenskaplikes nie die nodige ingewikkelde berekeninge kon doen om die kernbom te ontwerp nie. Dit is interessant om te weet dat dr. Von Neumann later bitterlik gekant was teen die bou van die waterstofbom. Dr. Oppenheimer het dieselfde gevoel en was van mening dat die rekenaar van meer nut as kernbomme is.

Intussen het rekenaars al hoe kleiner en kragtiger geword. In die begin was rekenaars groot masjiene – soos *ENIAC*. Met die koms van nuwe

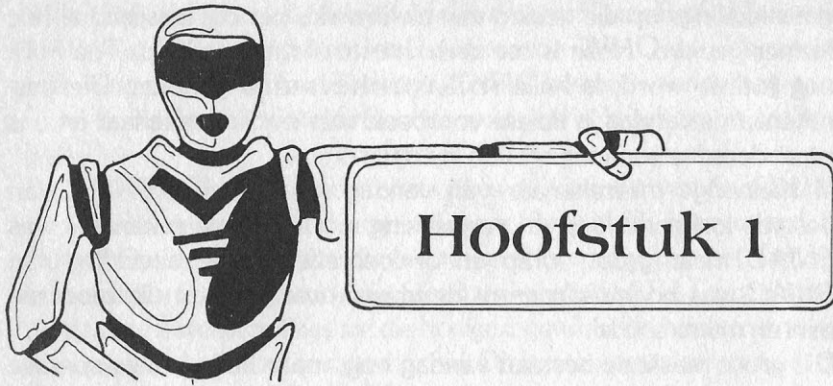


ENIAC was nie net groot nie, maar baie duur ook. Dit het destyds al meer as 'n half miljoen rand gekos om te bou. Elke sewe of agt minute moes duur onderdele vervang word en die instandhouding daarvan was dus baie duur.

ontwikkelings op die gebied van elektronika het die masjiene al hoe kleiner geword. Hulle is toe **minirekenaars** gedoop. Toe hulle nog kleiner word, is hulle **mikrorekenaars** genoem. Die tuisrekenaar is vandag 'n tipiese voorbeeld van 'n mikrorekenaar en ons praat deesdae sommer net van 'n **mikro**.

'n Kleinerige minirekenaar van vandag sal die reuse *ENIAC* van destyds loshande klop. In vergelyking met moderne rekenaars was *ENIAC* maar groot, lomp en ondoeltreffend. Die uitvinding van *ENIAC* was nogtans 'n groot deurbraak wat later tot die moderne mini en mikro sou lei.

Die groot masjiene bestaan vandag nog, maar hulle is baie spesiale apparate en staan bekend as die nuwe generasie supermasjiene. Dit sal natuurlik baie onregverdig wees om die mikro met hulle te vergelyk.



Basic is 'n baie eenvoudige rekenaartaal. Soos die naam aandui, is dit baie basies. Eintlik is dit 'n afkorting vir *Beginners All-purpose Symbolic Instruction Code.* Alle rekenaartale het ongelukkig groot, moeilike name en daarom gebruik ons afkortings daarvoor.

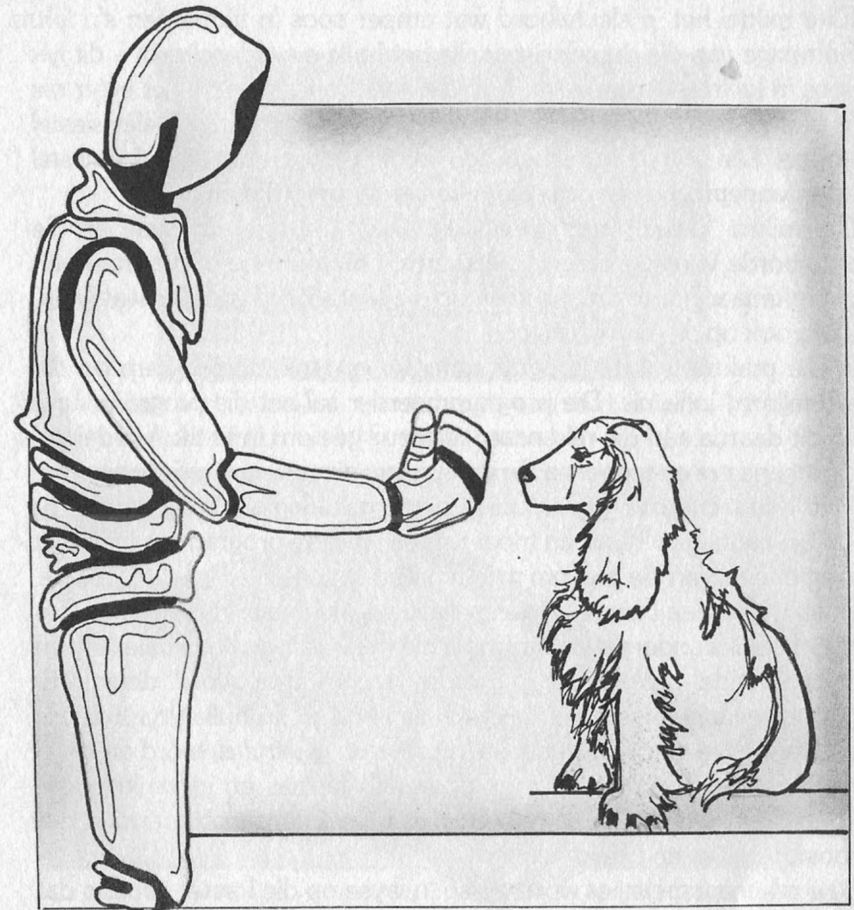
Basic is die heel eerste taal wat 'n programmeerder aanleer. Daarna sal hy of sy meer gevorderde tale soos *Fortran, Cobol* of *Assembler* aanleer.

Jy wonder seker nou wat 'n rekenaartaal presies is. Dis nie 'n taal soos Engels, Afrikaans of Duits nie, maar die manier en reëls waarmee ons met 'n rekenaar "praat". Daar is sekere instruksies of bevele wat ons kan gebruik. As ons hierdie instruksies korrek gee, sal die rekenaar dit verstaan en die bevele uitvoer. Onthou, 'n rekenaar is 'n dom masjien wat nie Afrikaans of enige ander mensetaal verstaan nie. Dis amper soos 'n hond vir wie jy toertjies aanleer. Jou hond verstaan nie Afrikaans nie, maar jy kan hom wel leer om op woorde soos sit, staan of lê te reageer. Net so verstaan 'n rekenaar net sekere woorde. Soos jou hond, sal dit pligsgetrou daarop reageer.

Daar is egter iets anders wat 'n rekenaar uniek maak: die geheue. Dit kan name, adresse, telefoonnommers of enige ander inligting dodelik akkuraat onthou.

Kom ons kyk eers hoe 'n eenvoudige tuisrekenaar, of 'n mikro lyk. Ons praat van 'n eenvoudige rekenaar, alhoewel dit eintlik 'n baie kragtige hulpmiddel is. Dit is byvoorbeeld honderde kere kragtiger as die duurste en beste sakrekenaar op die mark. Tog kos dit nie honderde kere duurder as 'n goeie sakrekenaar nie.

In 'n mikro is daar duisende klein onderdeeltjies wat elektroniese komponente genoem word. Gelukkig hoef die programmeerder nie te weet hoe hierdie tegnologiese wonderwerke werk nie, net soos die kaptein van 'n skip ook nie hoef te weet hoe die radar, sonar en ander ingewikkelde apparaat op sy skip werk nie. Hy het spesiaal opgeleide mense om na al hierdie tegniese dinge om te sien. Die programmeerder het weer ingenieurs, tegnici en operateurs wat saam met hom werk. Ons sien dus dat 'n programmeerder eintlik 'n baie belangrike persoon is. Dis nogal snaaks dat mense met 'n wiskundige aanleg in die ou dae as die beste programmeerders beskou is. Vandag



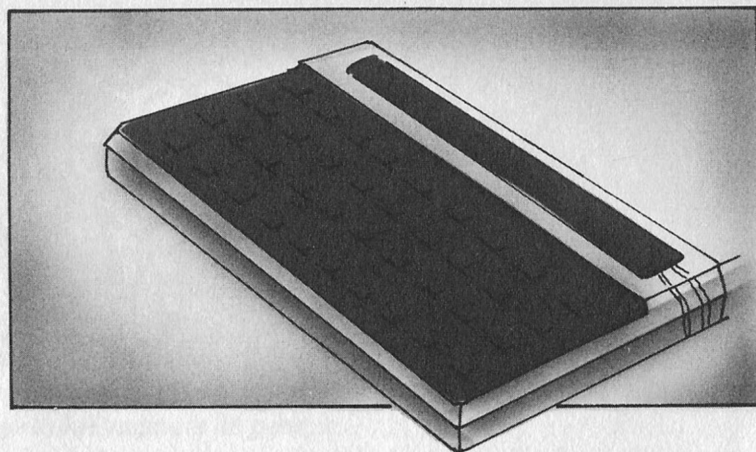
Soos 'n slim hond, reageer 'n rekenaar net op sekere woorde wat jy vir hom geleer het. Hierdie versameling woorde staan as die rekenaartaal bekend.

weet ons egter dat mense met 'n aanleg vir tale die beste programmeerders is!
Basic en tuisrekenaars is sinoniem, want alle tuisrekenaars verstaan *Basic*. Ongelukkig is daar vandag baie dialekte van *Basic*, met ander woorde verskillende variasies van die taal. Hierdie variasies of verskille is egter baie klein. Elke vervaardiger gebruik sy eie variasie vir sy tipe rekenaar. Die *Basic* wat ek jou in hierdie boek gaan leer is so elementêr dat dit op enige mikro sal werk. Dit is 'n baie eenvoudige vorm van die taal, maar daarmee sal julle kragtige programme kan skryf.
Elke mikro het 'n sleutelbord wat amper soos 'n tikmasjien s'n lyk. Sommige van die duurder modelle het hulle eie videoskerm - dit lyk soos 'n klein televisieskerm. Van die goedkoper mikro's het egter nie 'n skerm nie en om dit te gebruik moet jy dit aan 'n televisiestel koppel. Die verbinding is heel eenvoudig - jy trek net die televisiestel se antenneprop uit en druk die rekenaar se prop daarin.
Die mikro "praat" met jou op die skerm. Jy stel die vrae en die antwoorde verskyn dan op die skerm. Om jou vrae of program aan die rekenaar oor te dra, tik jy dit op die sleutelbord in. Alles wat jy tik, word ook op die skerm vertoon.
In die praktyk sal die programmeerder nie self die program op die sleutelbord intik nie. Die programmeerder sal net die program skryf en dit daarna aan die rekenaaroperateur gee om in te tik. Verder sal die ingenieurs en tegnici na die welsyn van die rekenaar omsien.
Met 'n tuisrekenaar is jy natuurlik nie so gelukkig om 'n operateur tot jou beskikking te hê nie en moet jy maar self jou programme intik. Dit is egter deel van die pret om met 'n mikro te speel.
Daar is deesdae baie rekenaarspeletjies beskikbaar. Hierdie speletjies is eintlik niks anders as programme nie (hulle is inderdaad baie lang en ingewikkelde programme). Hierdie programme word deur programmeerders geskryf en baie mense skryf dit in hulle vrye tyd om ekstra geld te verdien. Nadat so 'n program geskryf is, word dit op 'n magneetkaart ('n gewone musiekkasset) geplaas en in winkels verkoop. Daar is deesdae streng kopieregte op rekenaarprogramme, net soos op boeke en tydskrifte.
Baie rekenaarspeletjies word op so 'n wyse op die kasset geplaas dat dit feitlik onmoontlik is om die program op die kasset te lees. Programmeerders werk baie hard en lank aan hulle programme en hulle wil nie graag hê dat 'n ander programmeerder moet sien hoe hulle

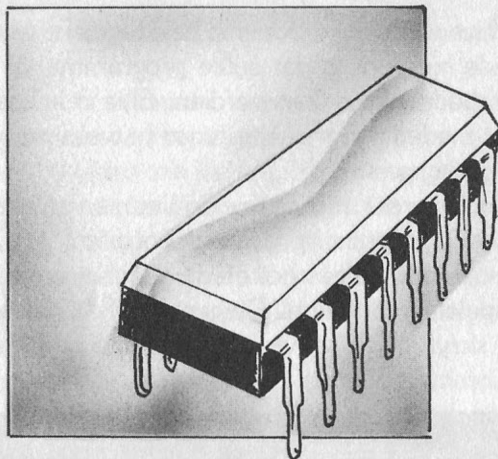
rekenaarspeletjies werk nie - trouens, baie tegnieke word dig geheim gehou. Die rede hiervoor is dat sulke programme vir die skrywers daarvan groot bedrae geld kan verdien. Elke rekenaarmaatskappy probeer om sy mededingers 'n stap voor te wees en so word nuwe programme elke dag geskryf.
'n Slim programmeerder met 'n bietjie kennis van elektronika kan egter so 'n geheime program op 'n kasset ontsyfer.
As jy hierdie boek deurgelees het en dit deeglik verstaan, kan jy self jou eie videospeletjies ontwerp. Diegene wat daarin belangstel kan gerus aan my skryf en ek sal vir julle volledige programme daaroor stuur.



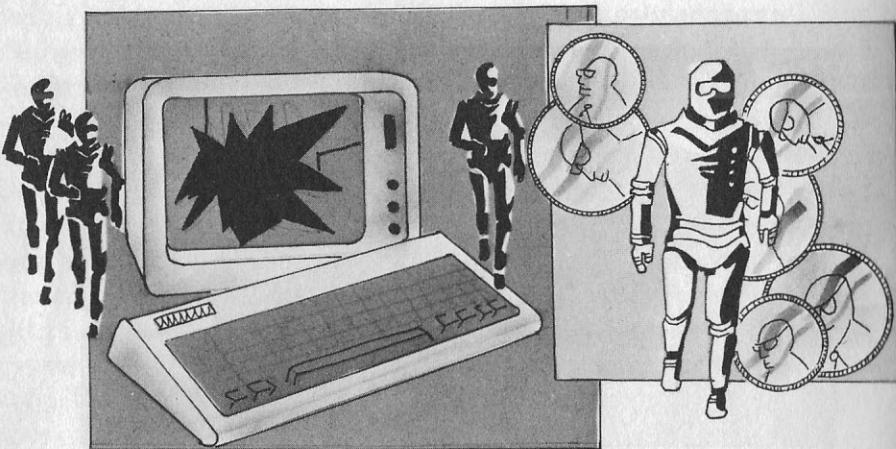
Mikrorekenaars met videoskerm. Hierdie is egter van die duurder modelle op die mark. Die moderne mikro is met 'n kleurskerm toegerus.



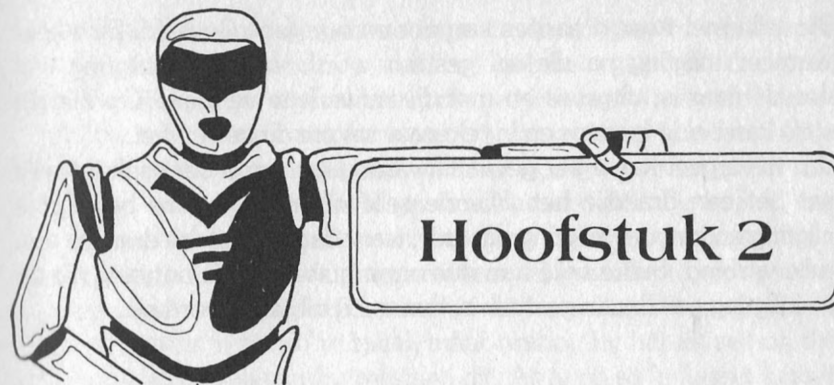
Een van die goedkoper en baie gewilde modelle wat nie met 'n videoskerm toegerus is nie. Dit is omtrent so groot soos 'n boek, maar is besonder kragtig en kan videospeletjies ook hanteer.



Die onderdele van 'n mikro lyk soos duisendpote, maar binne-in is duisende transistors en ander elektroniese komponente wat mikroskopies klein is. Die een wat julle hier sien is 'n gedeelte van die geheue.



Rekenaarspeletjies is deesdae baie gewild. Sommige van hulle bring vir die skrywers daarvan baie geld in die sak. 'n Mens kan ook byvoorbeeld skaak teen 'n mikro speel en jy sal vind dat dit 'n gedugte teenstander is.



Hoofstuk 2

Ons het vroeër van 'n rekenaar se geheue gepraat en gesê dat 'n rekenaar 'n baie goeie geheue het. Eintlik het 'n mens 'n baie beter geheue as 'n rekenaar, maar ons gebruik ons s'n net nie so doeltreffend nie.

Toe ingenieurs die rekenaar se geheue ontwerp het, het hulle die mens se geheue afgekyk, met die gevolg dat enige rekenaar se geheue baie soos 'n mens s'n werk.

In 'n mens se brein is daar miljoene klein selletjies wat almal saamwerk om dinge te onthou. Die selle in 'n mens se brein bestaan uit weefsel, terwyl 'n rekenaar s'n uit transistors en ander elektroniese komponente bestaan.

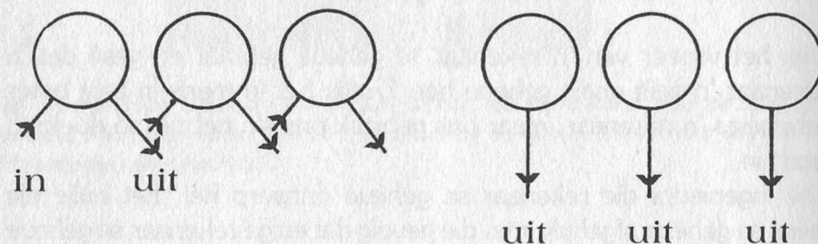
In 'n rekenaargeheue het elke sel een of twee fyn draadjies wat dit met die res van die rekenaar verbind. By die mens is hierdie draadjies natuurlik niks anders as senuwees nie. Deur hierdie draadjies en senuwees vloei 'n baie swak elektriese stroom. Dus kan 'n elektriese stroom na 'n geheuesel gestuur word en net so kan die selle weer elektriese strome uitstuur.

Die meeste geheueselle het twee draadjies – een om 'n elektriese stroom mee te vang en die ander om 'n stroom mee uit te stuur.

Wanneer 'n elektriese stroom na 'n sel gestuur word, word inligting in daardie sel gebêre of geberg. In die mens se geheue vind daar allerhande ingewikkelde chemiese reaksies plaas wanneer 'n sel 'n elektriese stroom ontvang. In 'n rekenaar word transistors aan- of afgeskakel wanneer dit gebeur.

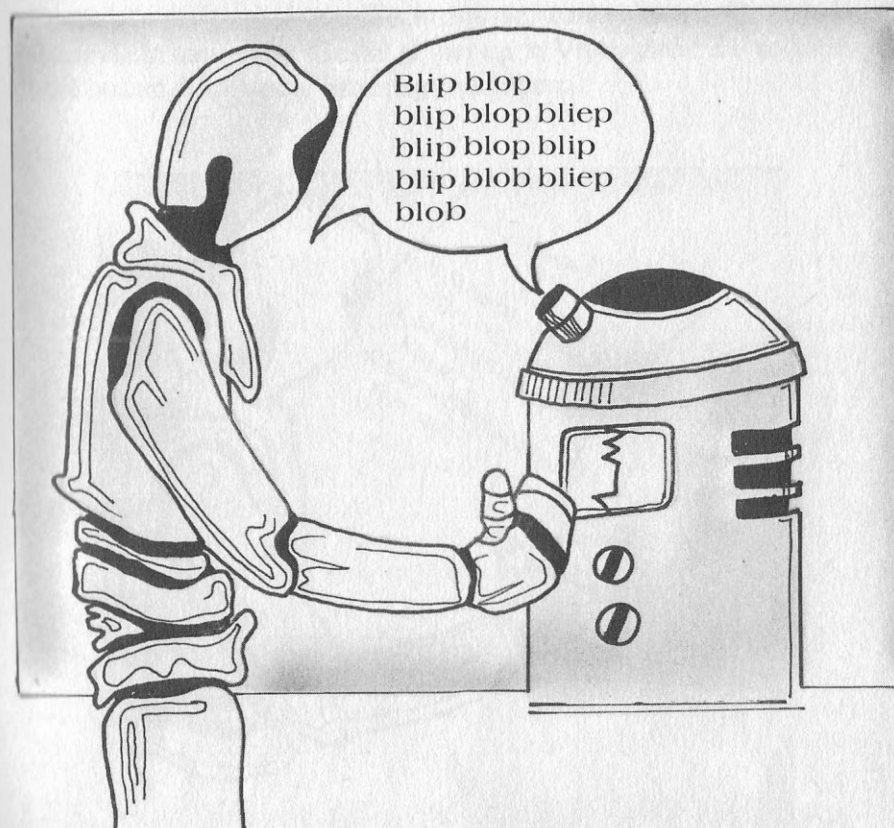
Die hele proses kom nou kortliks hierop neer: die elektriese stroom na die sel bêre inligting en as 'n stroom uitgestuur word vanaf 'n sel, word inligting uitgestuur.

Die selle met twee draadjies kan oor en oor gebruik word. Elke keer wanneer inligting na die sel gestuur word, word die inligting wat alreeds daar is, uitgewis en met die nuwe feite vervang. Op hierdie wyse kan beide 'n mens en 'n rekenaar sekere dinge vergeet.
Om te vergeet kan egter gevaarlik wees. Daarom is daar sekere selle wat net een draadjie het. Hierdie selle onthou net baie belangrike inligting en kan net inligting uitstuur, want daar is net een draadjie aan hulle verbind. Sulke selle kan dus nooit nuwe kennis ontvang nie en die inligting wat daarin gestuur is, kan nooit uitgewis word nie.



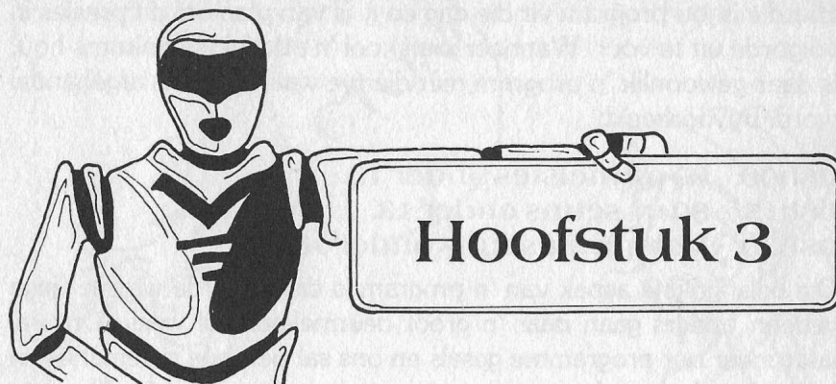
Verskillende soorte geheueselle

In 'n mens se geheue is daar inligting waarmee jy gebore word - **kennis** is miskien 'n beter woord. Jy hoef byvoorbeeld nooit te leer hoe om asem te haal nie, want daardie kennis is alreeds by geboorte in jou geheue. Dit is baie belangrik dat die geheueselle wat jou asemhaling en hartklop beheer, baie goed beskerm word. Daarom het hierdie selle net een senuwee. Hierdie area, of gedeelte van jou brein word die beskermde area genoem.
Die inligting of kennis in 'n rekenaar se beskermde geheue word in die fabriek daarin geplaas. Die persoon wat die rekenaar koop kan dit nooit uitwis nie, want dit word permanent in die geheue geplaas.
As jy nou jou mikro aanskakel, is die onbeskermde gedeelte van die geheue leeg. Dit sit nou vir jou en wag om iets daarin te plaas, soos 'n program byvoorbeeld. Die rekenaar is nou soos 'n pasgebore baba - dis heeltemal dom en weet niks nie. As jy nou 'n program intik, gaan dit wat jy tik na die geheue. Dan eers kan die mikro allerhande slim dinge begin doen.
Wanneer jy die rekenaar se krag afskakel, word alles in die onbeskermde gedeelte van die geheue uitgewis. As jy egter 'n waardevolle en lang program het wat jy weer wil gebruik, gaan dit natuurlik probleme skep. In so 'n geval kan jy die program op 'n magneetkasset bêre. Elke mikro het 'n manier waarop jy programme kan stoor. Die duurder modelle het 'n slapskyf waarop jy dit kan bêre, maar die goedkoper modelle gebruik 'n gewone kassetspeler. Jy koppel net eenvoudig enige kassetspeler aan die mikro en bêre jou program op kasset. As jy later dieselfde program weer wil gebruik, koppel jy weer die kassetspeler en speel die kasset vir die rekenaar. Dit mag snaaks klink om iets vir 'n mikro te speel, maar onthou hy het dit self op die magneetband geplaas en hy verstaan dit. As jy na so 'n kasset luister (doen dit gerus) sal jy allerhande snaakse *bliep-blop* geluide hoor. Luister gerus na die kasset van 'n rekenaarspeletjie en jy sal hoor hoe 'n rekenaar "praat".



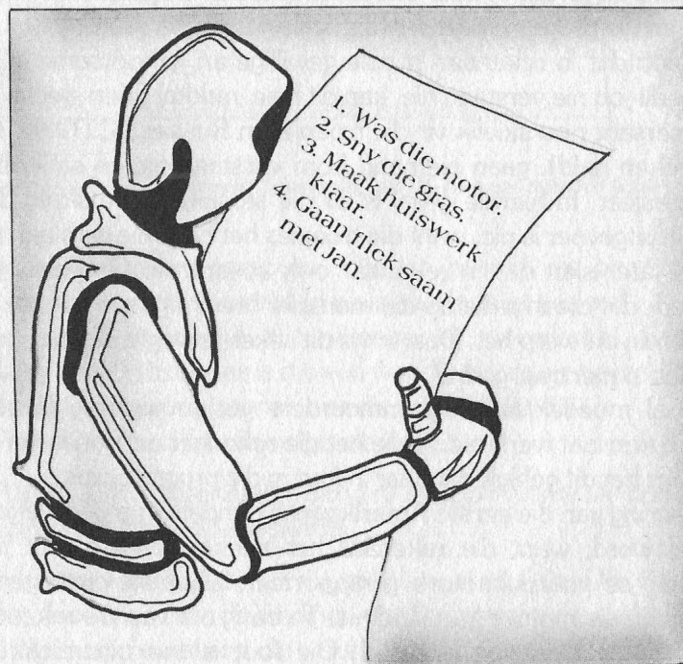
Ons het vroeër van 'n slapskyf gepraat. In die boek *Rekenaars* word dit baie volledig beskryf, maar ek sal net kortliks verduidelik hoe dit werk. Die slapskyf is 'n plastiese skyf wat 'n dun lagie magnetiese materiaal daarop het. Basies werk dit maar net soos 'n magneetband. Die voordeel van 'n slapskyf is dat dit baie vinnig werk. Waar dit 'n hele paar minute kan neem om 'n program op magneetband te berg of terug te lees, neem dit 'n slapskyf net 'n paar sekondes.

Sommige van die duurder modelle kan twee skywe gelyktydig hanteer. Die meganisme wat die skyf aandryf is gewoonlik in die mikro ingebou en daar is net 'n gleufie waarin jy die skyf steek. Sommige modelle het 'n afsonderlike skyfaandrywer.



Ons het tot dusver losweg van programme gepraat, maar nog nie mooi verduidelik wat dit presies is nie.

Eintlik is 'n program vir jou niks nuuts nie, want jy kom feitlik daagliks daarmee in aanraking. Gestel jy stel op 'n Vrydagaand die volgende lysie op van dinge wat jy Saterdag gaan doen:



Hierdie is jou program vir die dag en jy is van plan om dit presies in volgorde uit te voer. Wanneer jou skool 'n atletiekbyeenkoms hou, is daar gewoonlik 'n program met die tye waarop items afgehandel word. Byvoorbeeld:

08h00: 80 m, meisies onder 13.
08h15: 80 m, seuns onder 13.
08h30: Verspring, seuns onder 13.

Die belangrikste aspek van 'n program is die volgorde waarin dinge gebeur, anders gaan daar 'n groot deurmekaarspul wees. Ons sal later meer oor programme gesels en ons sal sien hoe sistematies en logies 'n rekenaar 'n program uitvoer deur slegs alles in die regte volgorde te doen.

Rekenaarinstruksies

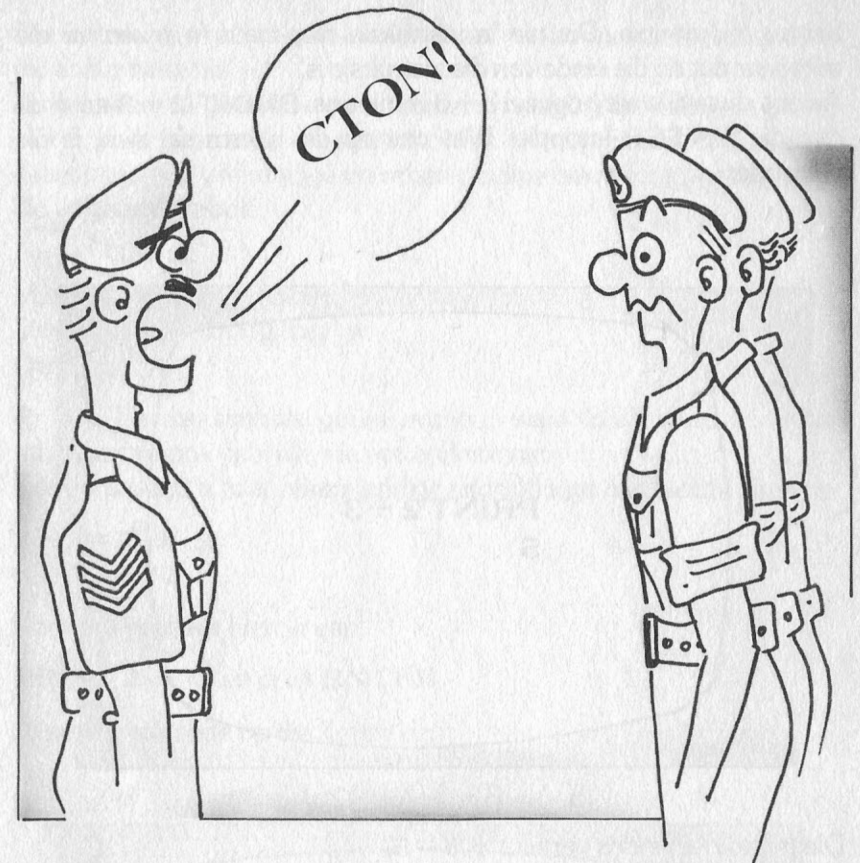
'n Instruksie is niks anders as 'n bevel nie. Wanneer 'n sersant aan die troepies die bevel: "Aandag!" gee, staan almal op aandag, want 'n bevel word uitgevoer sonder om teë te stribbel. 'n Rekenaar is soos 'n soldaat en die programmeerder (dis nou jy) is die sersant.

Jy sal vind dat 'n rekenaar 'n baie gewillige en gehoorsame slaaf is, maar as dit jou nie verstaan nie, kan dit baie nukkerig en steeks raak. As die sersant nou skielik vir die troepies in Russies "CTON!" skree (dit beteken halt!), gaan niemand hom verstaan nie en sal hulle bly voortmarsjeer. In hierdie geval is dit die sersant se eie skuld dat sy bevel nie uitgevoer is nie, want die troepies het hom nie verstaan nie.

Ons sal later sien dat 'n rekenaar ook so optree. Ons het vroeër opgemerk dat die ingenieurs die menslike brein afgekyk het toe hulle die blikbrein ontwerp het. Daarom is dit seker te wagte dat 'n rekenaar hom soos 'n mens sal gedra!

Ek het al moedelose programmeerders gesien wat nie 'n sekere program kon laat werk nie. Hulle het die rekenaar neuroties genoem, maar later het dit geblyk dat daar 'n fout in die program was.

Die lansering van die eerste Amerikaanse pendeltuig moes weke lank uitgestel word, want die rekenaar het aanhoudend 'n fout in die pendeltuig se vuurpylmotore gerapporteer, alhoewel die ingenieurs geen fout in die motore kon vind nie. Te bang om van die rekenaar te verskil, is die lansering uitgestel. Die fout is toe uiteindelik in 'n



As die sersant skielik begin Russies praat, sal die troepies hom nie verstaan nie.

rekenaarprogram opgespoor. Op een plek in die program het die programmeerder 'n komma in plaas van 'n punt geskryf!

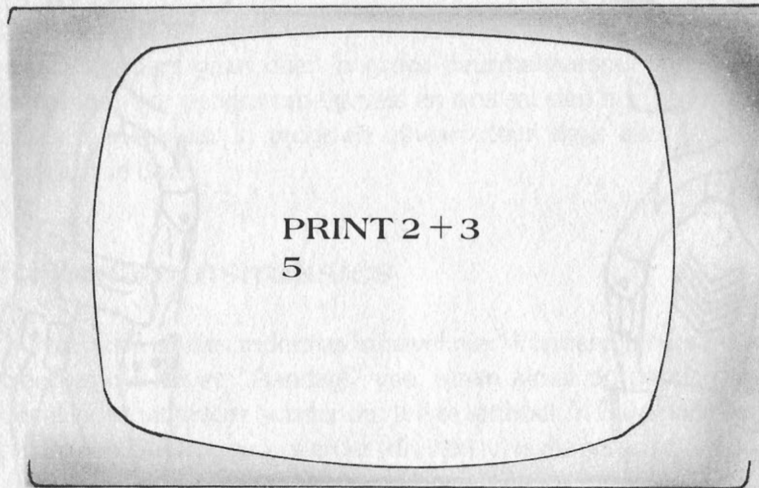
Kom ons kyk hoe lyk die rekenaar se woordeskat. Een van die belangrikste instruksies is dié een wat die rekenaar beveel om met jou te "praat". Hiervoor gebruik ons die woord PRINT.

As jy nou vir 'n mikro die volgende intik:

PRINT 2 + 3 beteken dit: *skryf of vertoon vir my die antwoord van die program 2 + 3 op die skerm.*

As ons net PRINT 2 + 3 tik, sal niks gebeur nie. Om die antwoord te kry moet ons eers die belangrikste knoppie op die sleutelbord druk. Dis die knoppie waarop die woord ENTER geskryf is. Elke rekenaar

het so 'n knoppie. Dis nie 'n instruksie nie, maar 'n teken vir die rekenaar dat dit die einde van die instruksie is.
As ons dus wil weet hoeveel 2 + 3 is, tik ons: PRINT 2 + 3 en druk dan die ENTER-knoppie. Wat ons op die skerm sal sien, is die volgende:



Die mikro het vir ons vertel 2 + 3 = 5.
Gestel jy het hierdie instruksie suksesvol uitgevoer. Wat nou? Die mikro staan nou geduldig en wag vir jou volgende instruksie.
Gestel jy het 'n tikfout gemaak en die woord PRINK getik. Die rekenaar sal dit nie verstaan nie, want daar bestaan nie so 'n rekenaarinstruksie nie. In so 'n geval gaan dit protesteer en 'n foutboodskap flits. Verskillende mikro-modelle het verskillende foutboodskappe. 'n Tipiese een is die sarkastiese

NONSENSE IN BASIC.

Terloops, in *Basic* word spasies tussen uitdrukkings geïgnoreer. Die volgende drie is presies dieselfde:

(*a*) PRINT 2 + 3
(*b*) PRINT 2+3
(*c*) PRINT2+3

Die instruksie in (*c*) is ook korrek, maar nie so maklik leesbaar soos die ander twee nie.
Jy het seker al gewonder hoe vermenigvuldiging en deling uitgevoer word, want nêrens op die sleutelbord is 'n knoppie met 'n deelteken daarop nie. Vir wiskundige en rekenkundige bewerkings gebruik ons die volgende simbole:

Plus +
Minus —
Vermenigvuldiging *
Deling /

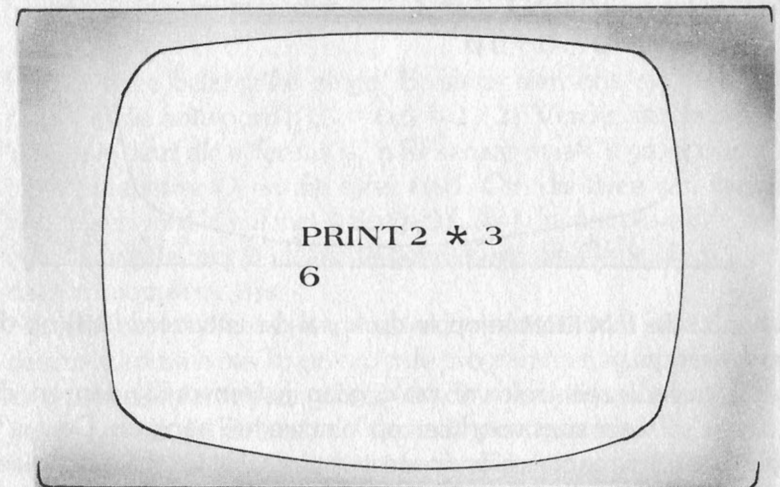
Jy kan hierdie simbole gerus onthou, want hulle word in ander rekenaartale ook gebruik, nie net in *Basic* nie.
Kom ons skryf 'n paar rekenkundige uitdrukkings met hierdie simbole.

2×3=2 * 3
6÷2=6/2

Kom ons probeer hierdie een:

PRINT 2 * 3 en druk ENTER

Jy sal die volgende op die skerm sien:

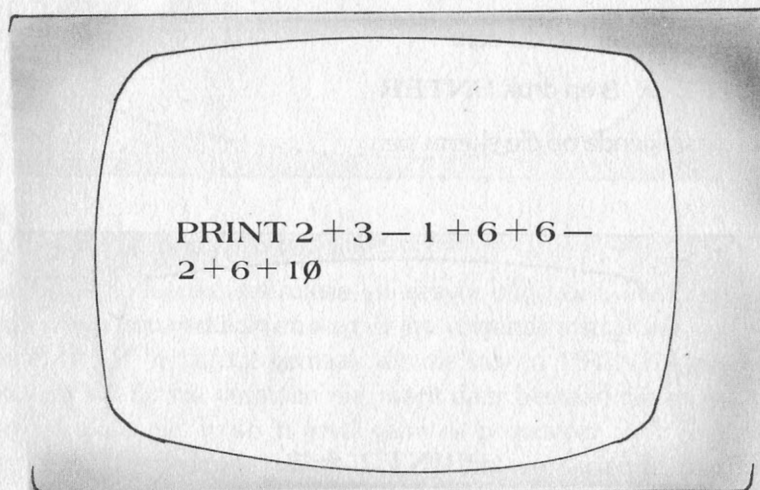


Miskien wonder jy nog oor hierdie knoppie met die woord ENTER

daarop. Terwyl ons besig is om die instruksie te tik, weet die rekenaar nog nie daarvan nie, alhoewel dit al op die skerm verskyn. Eers wanneer jy ENTER druk, is dit vir die rekenaar die teken om dit wat nou op die skerm staan, te lees en die opdrag uit te voer. Daar is nog ander redes daarvoor ook. Die meeste skerms is maar redelik klein – sommige van hulle kan net 40 kolomme neem, ander soorte tot soveel soos 80. As jy nou 'n lang instruksie tik, gaan jy baie gou aan die einde van die reël kom. In so 'n geval hou jy maar net aan tik. Jy sal sien dat die instruksie êrens afgekap word en op die volgende reël voortgaan. Die rekenaar beskou elke instruksie as een lang reël wat eers eindig wanneer jy ENTER druk. Dit is maar net dat die skerm te klein is om alles as een reël te vertoon.
'n Voorbeeld sal dit duideliker maak:

PRINT 2 + 3 − 1 + 6 + 6 − 2 + 6 + 10

Op die skerm sal dit so lyk:

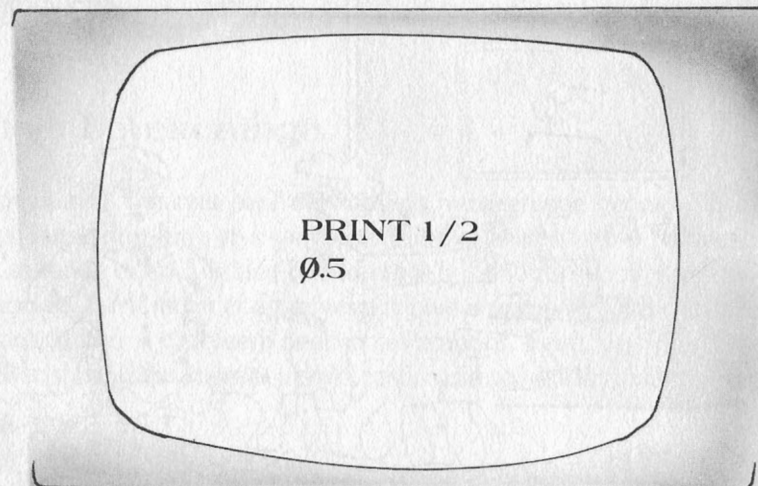


As jy nou die ENTER-knoppie druk, sal die antwoord (3Ø) op die skerm verskyn.
As die tweede reël ook vol raak, gaan jy eenvoudig aan en die rekenaar sal weer soos voorheen op 'n nuwe reël aangaan. Daar is 'n beperking op die aantal reëls vir een instruksie. Dit verskil van masjien tot masjien, maar die meeste mikro's kan tot 20 reëls per instruksie aanvaar.

Ons kan met desimale breuke ook werk. Hier is 'n belangrike ding wat jy altyd in gedagte moet hou: rekenaars word in oorsese lande vervaardig waar die desimale punt (.) in plaas van die desimale komma gebruik word. Tik nou die volgende in:

PRINT 1/2
(ons vra: "Hoeveel is een gedeel deur twee?")

As jy nou ENTER druk, sal die skerm so lyk:

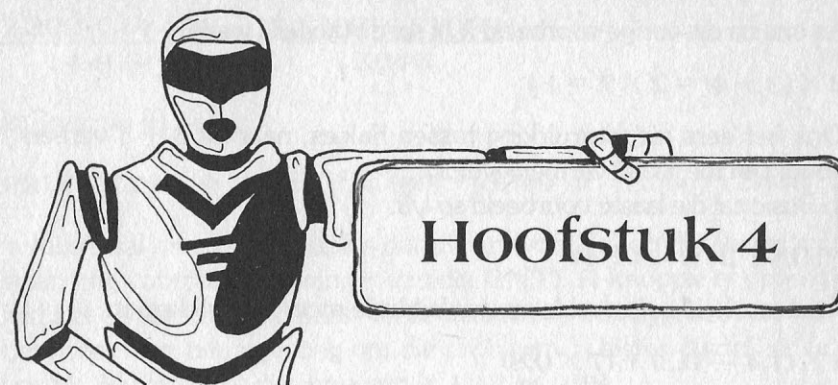


Hier is twee belangrike dinge. Eerstens sien ons die desimale punt in die antwoord (0,5 = 0.5 = 1/2). Verder sien ons dat daar 'n strepie deur die syfer nul is. 'n Rekenaar maak 'n groot onderskeid tussen die letter O en die syfer nul. Om die twee van mekaar te onderskei word 'n nul met Ø aangedui. As jy 'n instruksie intik moet jy ook daarop let om Ø vir nul te gebruik, en nie O nie. Jy sal sien dat daar 'n knoppie vir Ø is.
Net so moet jy ook onthou om altyd 'n desimale punt in te tik en nie 'n desimale komma nie. In gevorderde programmering word die komma vir iets heeltemal anders gebruik en die rekenaar sal 'n foutboodskap gee as jy dit in die plek van die desimale punt gebruik.
Miskien het jy al gewonder waarom ons die woord PRINT gebruik. Dit beteken tog immers DRUK en 'n televisieskerm kan nie dinge DRUK nie. Dit is wat ons 'n *historiese fout* noem. Jare gelede toe die

eerste rekenaars in gebruik geneem is, was videoskerms nog skaars en baie duur. Toe is elektriese tikmasjiene gebruik om met 'n rekenaar te praat. Later het lyndrukkers en videoskerms meer in gebruik geraak. As 'n mikro met 'n drukker toegerus is, gebruik ons die instruksie **LPRINT** as ons die resultate op die drukker uitgedruk wil hê. Daar is egter nie baie mikro's wat met drukkers toegerus is nie en daarom sal ons **LPRINT** nie verder bespreek nie.



"Ja, die nuwe rekenaar is baie gevoelig vir sarkastiese opmerkings!"



Hoofstuk 4

Meer berekeninge

Tot dusver het ons baie eenvoudige berekeninge gedoen. 'n Mikro kan egter baie lang en ingewikkelde berekeninge binne 'n fraksie van 'n sekonde doen. Vir lang berekeninge is dit soms handig om hakies te gebruik. 'n Mens moet egter versigtig wees wanneer jy hakies gebruik, want dit kan 'n probleem heeltemal verander. Teen hierdie tyd weet jy seker al dat daar 'n groot verskil tussen die volgende probleme is:

(a) $2 \times 3 + 4$
(b) $2 \times (3 + 4)$.

(a) $2 \times 3 + 4 = 6 + 4 = 10$
(b) $2 \times (3 + 4) = 2 \times 7 = 14$

Wanneer ons rekenkundige bewerkings uitvoer, is daar 'n sekere volgorde van belangrikheid. So is vermenigvuldiging en deling belangriker as optel en aftrek. Daarom word vermenigvuldiging en deling **voor** optel en aftrek uitgevoer. Aan die ander kant is hakies weer belangriker as vermenigvuldiging en deling. As daar hakies in 'n uitdrukking is, word die uitdrukking tussen die hakies eers vereenvoudig en daarna word vermenigvuldiging, deling, optel en aftrek gedoen. 'n Rekenaar volg ook hierdie prosedure en ons gee dit vir jou in 'n tabelvorm:

1. **Vereenvoudig uitdrukkings tussen hakies.**
2. **Vermenigvuldig en deel.**
3. **Tel op en trek af.**

As ons na die vorige voorbeeld kyk sal dit duidelik wees:

$2 \times (3+4) = 2 \times 7 = 14$

Ons het eers die uitdrukking tussen hakies, naamlik 3 + 4 vereenvoudig en toe met 2 vermenigvuldig.
In *Basic* sal die laaste voorbeeld so lyk:

```
PRINT 2 * (3 + 4)
```

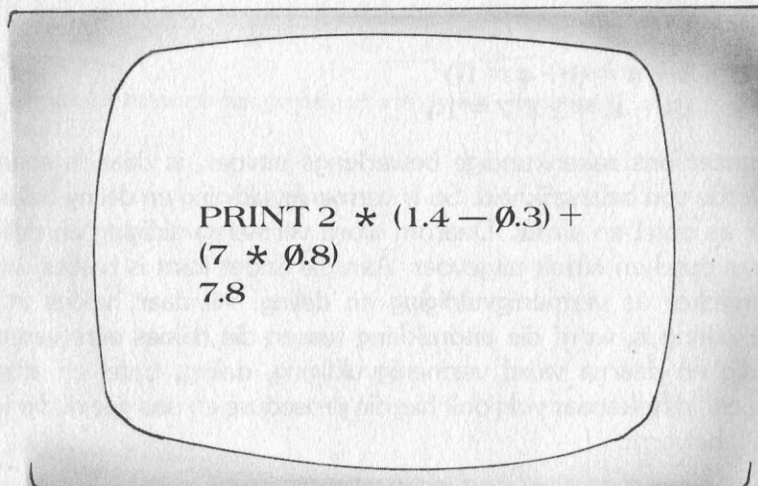
Kom ons kyk na 'n probleem wat 'n bietjie meer ingewikkeld is:

$2 \times (1{,}4 - 0{,}3) + (7 \times 0{,}8)$

In *Basic* lyk dit so:

```
PRINT 2 * (1.4 — Ø.3) + (7 * Ø.8)
```

Dit mag 'n bietjie vreemd lyk, maar as jy onthou wat ons oor die desimale punt en die syfer nul (Ø) gesê het, sal dit duidelik wees.
Kom ons gee die laaste voorbeeld vir 'n mikro en kyk hoe die skerm lyk.
Die rekenaar sê die antwoord is 7,8. Werk dit self uit en kyk of jy daarmee saamstem.



Die volgende probleem is heelwat ingewikkelder en dit sal jou lank neem om dit self uit te werk:

$$\frac{7{,}209 \times 14{,}491}{14{,}409} + \frac{6{,}309 \times 13{,}999}{12{,}646}$$

In *Basic* sal dit só lyk:

```
PRINT (7.2Ø9 * 14.491 / 14.4Ø9) + (6.3Ø9 * 13.999 / 12.646)
```

'n Mikro sal dit so vinnig oplos dat die antwoord al op die skerm sal staan nog voordat jy jou vinger van die ENTER-knoppie af opgelig het! Let op dat ons die hakies hierbo netsowel kon weggelaat het - ek het hulle maar net ingevoeg om die probleem 'n bietjie duideliker te laat lyk. My mikro sê die antwoord is: 14,234 028.
Vervolgens gaan ons kyk hoe 'n mikro **getalle** kan onthou. Dit bring ons by die volgende woord, naamlik LET.
Alle mense het name, want daar is baie mense op aarde en dit sou maar 'n deurmekaarspul gewees het as ons nie name gehad het nie. As jou klasonderwyser sê: "Kom haal jou boek, Jan de Wet," dan weet almal in die klas met wie die onderwyser praat.
As 'n rekenaar 'n paar duisend getalle in sy geheue het, word aan elke getal **'n naam** toegeken. Dit mag verspot klink, maar op hierdie wyse is dit later baie maklik om getalle uit die geheue te kry.
Name word deur die programmeerder toegeken en daar is 'n paar reëls wat hier geld. **'n Naam moet altyd met 'n letter van die alfabet begin.** Daar is ook 'n beperking op die lengte van 'n naam, maar dit verskil van rekenaar tot rekenaar. Van die ouer masjiene het slegs twee letters per naam toegelaat, maar van die nuwe mikro's aanvaar tot tien letters per naam.
'n Naam soos 2JAN is ongeldig, want dit begin nie met 'n letter van die alfabet nie. JAN, KOOS, PIET, SAREL, JANNEMAN, JAN3, SAREL6 is almal geldige name. Ons sien dus dat daar 'n groot verskeidenheid van name is waaruit die programmeerder kan kies.
Enkel letters soos A,B,C,D ensovoorts is natuurlik ook geldige name. As ons nou wil hê dat die rekenaar 'n sekere getal moet onthou, kies ons eers 'n naam daarvoor en gebruik die LET-instruksie. Gestel die getal is 23 en ons besluit sy naam moet JAN wees. Dan sê ons:

```
LET JAN = 23
```

So eenvoudig soos dit!
Maar hoe weet ons nou verseker dat die mikro die getal werklik

onthou het? Dis baie maklik om uit te vind. Ons tik net: PRINT JAN.

Die rekenaar se geheue is in klein hokkies ingedeel - amper soos die rye en rye posbussies wat jy voor 'n woonstelblok se ingang sien. As jy vir die rekenaar sê LET JAN = 23, gaan soek hy die eerste leë posbussie in sy geheue en skryf JAN daarop. Dan sit hy die getal 23 daarin. As jy nou vir hom sê PRINT JAN, gaan soek hy in sy geheue na die posbussie met JAN daarop en lees die getal daarin en skryf die getal op die skerm. Die getal 23 bly nog steeds in die geheue - dis baie belangrik om dit te onthou.

Ons kan 'n getal in die geheue verander ook. Tot dusver het ons die getal 23 in die geheue. Gestel ons wil dit na 29 verander. Dan sê ons net weer: LET JAN = 29. Elke keer wanneer ons vir die rekenaar 'n naam gee, soek hy blitsvinnig deur sy geheue om te sien of daar nie reeds 'n posbussie met daardie naam is nie. Indien daar nie een is nie, maak hy 'n nuwe een oop. Indien daar reeds so 'n posbussie is, vervang hy net die ou getal daarin met die nuwe een. Kom ons probeer om nou meer as een naam op 'n keer te gebruik:

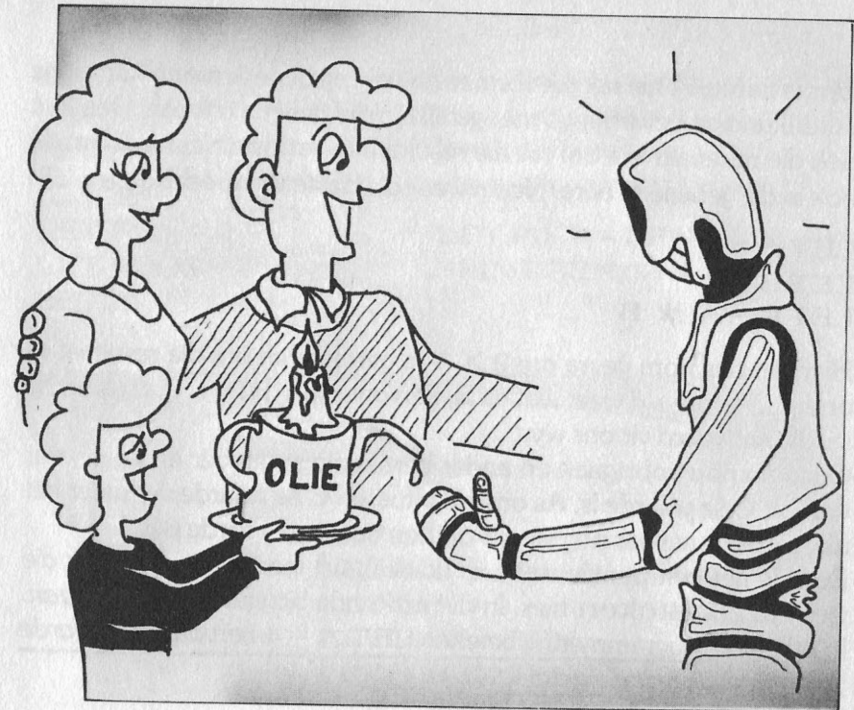
```
LET JAN = 33
LET KOOS = 45
LET KOSIE = 2
PRINT JAN
PRINT KOOS
PRINT KOSIE
```

Die rekenaar sal nou die getalle 33, 45 en 2 vertoon.

Op hierdie wyse kan ons duisende getalle in die geheue berg. Die gedeelte van die geheue waar dit plaasvind is natuurlik die onbeskermde geheue. Die beskermde geheue het ook posbussies, maar hulle het nie deurtjies nie. Hulle het glasdakkies waardeur die rekenaar kan lees wat binne is. Hy kan glad nie die inhoud daarvan verander nie. Onthou jy wat ons vroeër oor die beskermde geheue gesê het? Ons het die rekenaar se brein met dié van die mens vergelyk.

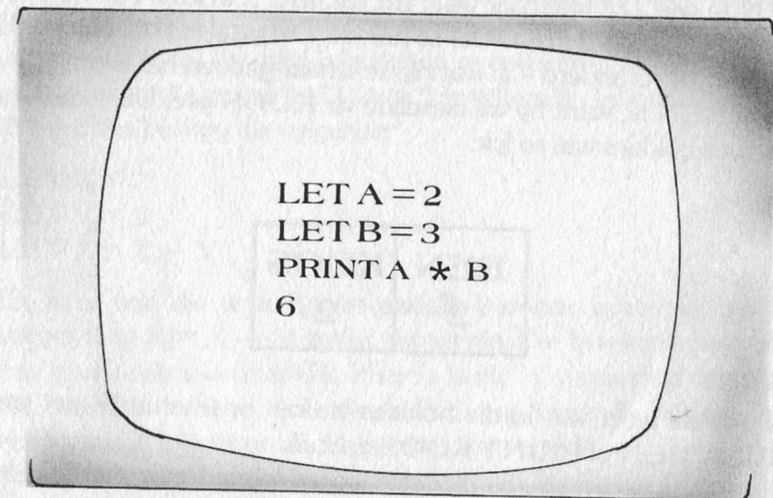
Noudat ons die mikro so ver gekry het om getalle te onthou, kan ons hom ook vra om **berekeninge** met die getalle in sy geheue te doen. Kyk na die volgende:

```
LET A = 2
LET B = 3
PRINT A * B
```

"Veels geluk, liewe Mikro!"

Hier vra ons hom in die laaste reël om die **produk** van A en B te bereken, dit wil sê 2 × 3. Op die skerm sal dit soos volg daar uitsien:



Ons is natuurlik nie net tot vermenigvuldiging beperk nie en kan enige rekenkundige bewerkings met getalle in die geheue uitvoer. Ons kan ook die rekenaar vra om nie die resultate te vertoon nie, maar om dit ook in die geheue te bêre. Neem weer die laaste voorbeeld:

```
LET A = 2
LET B = 3
LET C = A * B
```

Hier het ons hom gevra om 2 × 3 te bereken en om die resultaat te onthou. As ons wil weet wat dit is, kan ons net **PRINT C** druk en hy sal die antwoord vir ons wys.
Ons kan nou voortgaan en ander berekeninge uitvoer en later weer vra wat C se waarde is. As ons nie intussen C se waarde verander het nie, sal die rekenaar nog steeds onthou dat C se waarde 6 is.
Daar is net een puntjie wat ons duidelik wil maak. Dit gaan oor die **gelykheidsteken** hier. In die wiskunde beteken = *is gelyk aan.* In rekenaarprogrammering beteken LET A = 4 *vervang die waarde van hokkie A met die getal 4.*
Kyk of jy die volgende een verstaan:

```
LET BEN = 2
LET KOOS = BEN
```

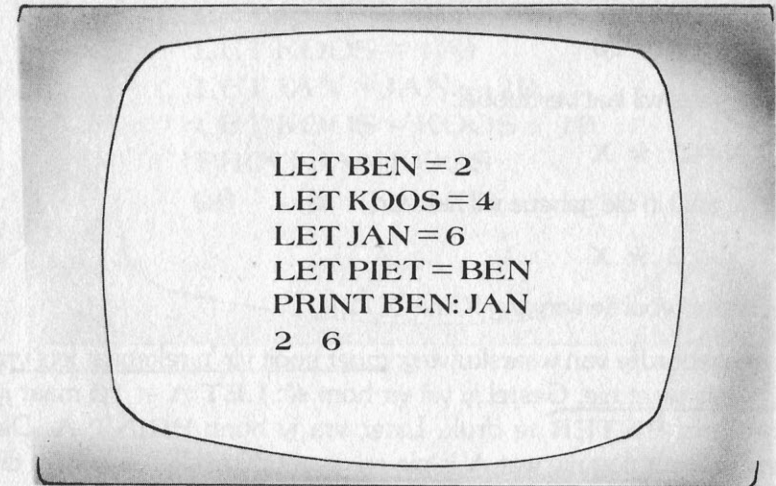
Wat hier gebeur is dat die rekenaar eers die getal 2 in BEN se hokkie plaas. Daarna vra ons hom om KOOS dieselfde waarde as BEN te gee. Die rekenaar weet dat die getal 2 in BEN se hokkie is en sit ook 'n getal 2 in KOOS se hokkie. Die getal in BEN se hokkie bly egter onveranderd - al wat die rekenaar gedoen het is om te kyk wat in BEN is, want hy wil dieselfde vir KOOS gee. Uiteindelik het ons twee hokkies wat so lyk:

BEN	KOOS
2	2

As ons wil weet wat in die hokkies is, kan ons natuurlik net weer PRINT BEN of PRINT KOOS gebruik.
Daar is iets waarmee ons die lewe vir onsself 'n bietjie makliker kan

maak. Ons kan meer as een naam in 'n PRINT-instruksie spesifiseer, byvoorbeeld PRINT BEN; KOOS. In so 'n geval sal die rekenaar die twee getalle langs mekaar op dieselfde reël vertoon.
As ons nou die volgende stel instruksies intik, sal ons skerm so lyk:

```
LET BEN = 2          LET PIET = BEN
LET KOOS = 4         PRINT BEN; JAN
LET JAN = 6
```



Jy onthou natuurlik nog om aan die einde van elke instruksie ENTER te druk!
Jy sal onthou dat die gelykheidsteken se betekenis hier *word vervang met* beteken. As ons sê LET A = 2 beteken dit *A word vervang met 2.* Kom ons beskou die volgende:

```
LET X = 7
LET Y = 3
LET X = X — Y
```

Ek hoor hoe die wiskundiges dadelik beswaar aanteken, want die vergelyking X = X — Y maak nie sin nie. Die betekenis hiervan het ons eintlik reeds verduidelik. Hierdie is nie 'n wiskundige vergelyking nie. LET X = X — Y beteken: *Vervang X met X-Y.* Voordat die rekenaar dit kan doen moet hy eers die waarde van X — Y gaan bereken en dan kan hy hierdie waarde aan X toeken.

'n Waardevolle tegniek is die volgende:
Gestel jy wil 'n getal in die geheue met een verminder. Dit doen jy soos volg:

```
LET X = X — 1.
```

Maklik, is dit nie? As jy die getal met een wil vermeerder, skryf jy:

```
LET X = X + 1.
```

As jy dit met 10 wil vermeerder, gebruik jy:

```
LET X = X + 1Ø
```

As jy 'n getal wil laat verdubbel:

```
LET X = 2 * X
```

As jy 'n getal in die geheue wil halveer:

```
LET X = 2 * X
```

(met ander woorde vervang X met X/2)

Net een woordjie van waarskuwing: moet nooit vir 'n rekenaar iets vra wat hy nie weet nie. Gestel jy wil vir hom sê: LET A = 1Ø maar jy vergeet om ENTER te druk. Later vra jy hom: PRINT A. Die rekenaar weet nog nie wat A is nie en gaan op jou skel as jy hom dit vra!
Kom ons kyk na 'n praktiese situasie en probeer om dit in rekenaartaal te beskryf.
Jan en Koos sit en dobbel en elkeen begin met R100. Met die eerste spel verloor Jan R10. Hoeveel het elkeen nou?
Onthou Jan en Koos is wettige name in *Basic* en ons skryf dit soos volg neer:

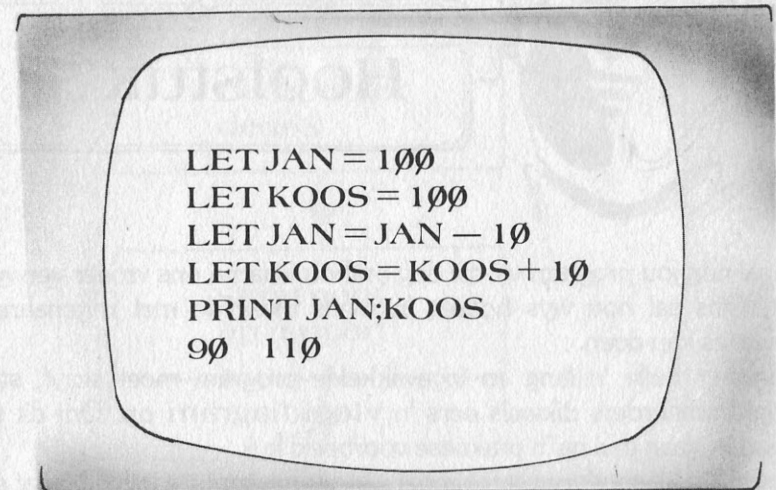
```
LET JAN = 1ØØ
LET KOOS = 1ØØ
```

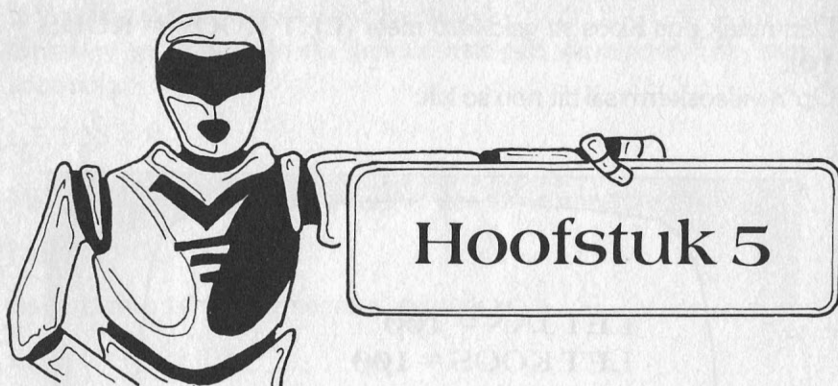
Die spel het nog nie begin nie en elkeen het R100. Jan verloor R10, wat beteken dat Koos R10 wen.

```
LET JAN = JAN — 1Ø
LET KOOS = KOOS + 1Ø
```

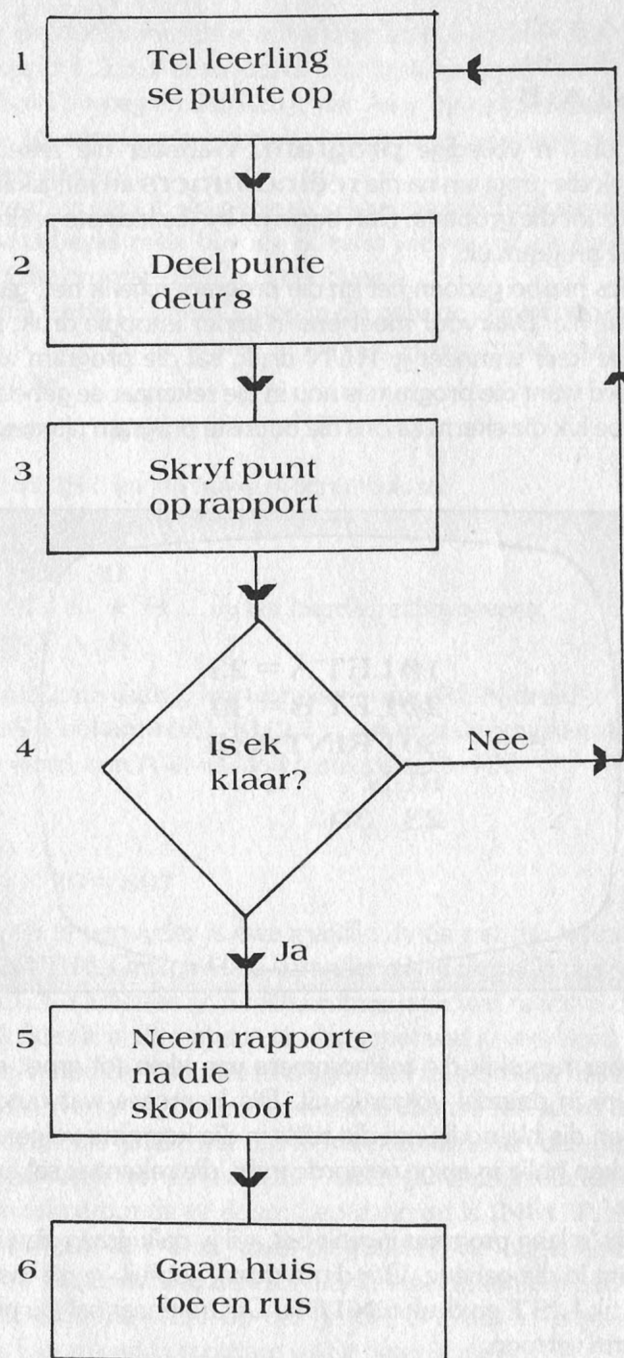
Eers maak ons Jan se geld R10 minder (LET JAN = JAN — 1Ø).

Dan maak ons Koos se geld R10 meer (LET KOOS = KOOS + 1Ø).
Op 'n videoskerm sal dit nou so lyk:



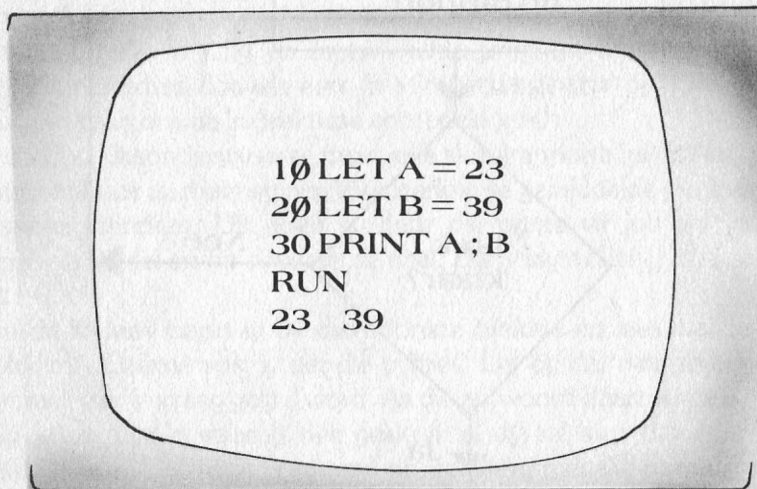


Jy sal nog jou program vir die dag onthou waarna ons vroeër verwys het. Ons sal nou wys hoedat 'n mens dieselfde met rekenaarinstruksies kan doen.
Wanneer hulle 'n lang en ingewikkelde program moet skryf, stel programmeerders dikwels eers 'n **vloeidiagram** op. Om dit te illustreer gaan ons na 'n praktiese voorbeeld kyk.
Gestel jou klasonderwyseres moet julle skoolrapporte invul. Nadat sy daarmee klaar is, moet sy nog elke leerling se gemiddelde punt vir die eksamen bereken. Dit doen sy deur die punte vir jou agt vakke bymekaar te tel en dit deur agt te deel. Die **vloeidiagram** sal so lyk:
Om dit te lees begin jy by die boonste blokkie en lees wat in die blokkie is. Daarna volg jy net die pyltjies. Let op dat daar in blokkie nommer vier 'n vraag gestel word. As die antwoord daarop *nee* is, dan volg jy die pyltjie waarby *nee* geskryf is. Jy sal sien dat daar ses blokkies is en dat hulle nommers ooreenstem met die nommers op die program wat ons later daaroor kan skryf. Onthou jou program vir die dag het ook nommers gehad. In 'n *Basic*-program gebruik ons ook **nommers**.
Al die instruksies wat ons sover ingetik het, het natuurlik verlore gegaan, want ons het nie die rekenaar gevra om dit te onthou nie. Ons het hom wel gevra om sekere getalle te onthou. **As ons die rekenaar vra om die stel instruksies te onthou, word dit 'n program. Om 'n instruksie te laat onthou, plaas ons net 'n reëlnommer voor die instruksie.** Die reëlnommers moet almal verskillend van mekaar wees, maar hulle hoef nie op mekaar te volg nie.



```
1Ø LET A = 23
2Ø LET B = 39
3Ø PRINT A ; B
```

Hier het ons 'n volledige **program**. Wanneer die rekenaar dit uitvoer, kyk die program na die **reëlnommers** en rangskik dit van die kleinste tot die grootste. Dan begin hy by die kleinste reëlnommer en voer die program uit.
As ons alles hierbo gedoen het en die program ingetik het, gaan daar niks gebeur nie. Daarvoor moet ons 'n ander knoppie druk, naamlik RUN. Elke keer wanneer jy RUN druk, sal die program weer uitgevoer word want die program is nou in die rekenaar se geheue. Kom ons kyk hoe lyk die skerm as ons die boonste program uitvoer.



Die rekenaar rangskik die reëlnommers van klein tot groot en voer die program in daardie volgorde uit. Dis 'n proses wat outomaties plaasvind en dis nie nodig om die reëls in die korrekte volgorde in te tik nie. Jy kan hulle in enige volgorde intik, die rekenaar sal hulle self rangskik.
Wanneer jy 'n lang program ingetik het, wil jy dalk graag sien hoe lyk die program in die geheue. Om dit te doen, gebruik jy die instruksie LIST. Jy tik LIST en druk ENTER. Die rekenaar sal jou program op die skerm vertoon.

Jy kon die reëlnommers op mekaar laat volg het indien jy wou, byvoorbeeld 1,2,3,4 ensovoorts. Die nadeel is egter dat jy later dalk nog reëls wil byvoeg en dan kan jy nie. As jy byvoorbeeld die nommers 1Ø, 2Ø 3Ø ensovoorts gebruik, kan jy later nog nege reëls tussen enige twee invoeg.
As 'n program eers in die geheue is, kan jy enigiets daarmee doen. Jy kan byvoorbeeld reëls byvoeg of reëls uithaal, of die hele program uitwis, of die program nog 'n keer uitvoer.
Gestel die laaste program is nog in die geheue. Jy wil nou graag weet hoeveel is A × B. Jy besluit jy wil nog 'n reël tussen reëls 20 en 30 invoeg en tik:

```
25 PRINT A * B
```

Druk nou LIST en jou nuwe program lyk so:

```
1Ø LET A = 23
2Ø LET B = 39
25 PRINT A * B ... Jy het hierdie reël bygevoeg.
3Ø PRINT A ; B
```

Watter resultate gaan jy kry wanneer jy nou RUN druk?
Reël 25 is bokant **reël 3Ø** (25   3Ø) en daarom gaan A × B eers vertoon word, dan A en B. Jou resultaat sal so lyk:

897
23 39
want 23 × 39 = 897

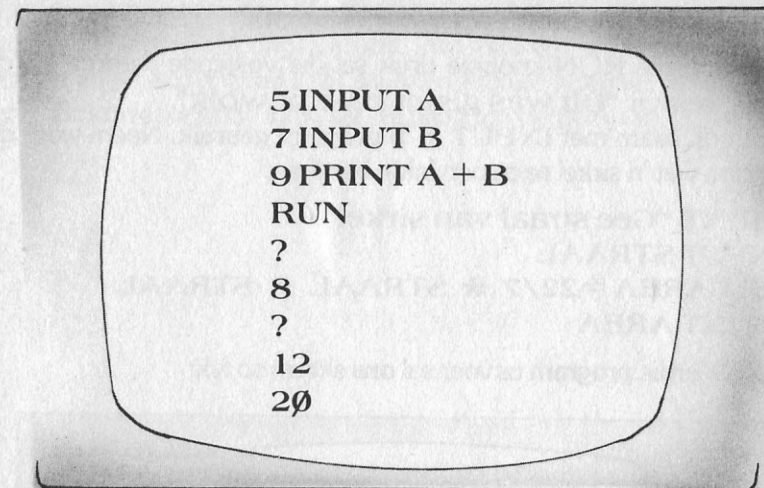
Om 'n reël te verwyder is ewe maklik. Jy tik net die reëlnommer en druk ENTER. Om 'n reël te verander, tik jy die reël soos jy dit wil hê en druk ENTER. As jy 'n reëlnommer intik wat reeds in die geheue is, word die reël in die geheue vervang met wat jy vervolgens gaan tik. As jy net 'n nommer tik met niks agterna nie, word die bestaande reël met niks vervang nie, met ander woorde daardie reël word uitgewis.
Wanneer jy baie getalle het om vir 'n rekenaar in te voer, gaan dit baie uitputtend wees om dit met LET-stellings te doen. Gelukkig is daar ander maniere om dit te doen. Een daarvan is INPUT. As jy vir 'n mikro sê: 1Ø INPUT A, gaan dit geduldig vir jou sit en wag om 'n getal in te tik. Sommige mikro's flits 'n vraagteken om aan te dui dat dit vir 'n getal wag. Let op dat jy INPUT net in 'n program kan gebruik. Die volgende program sal dit beter illustreer.



"Ek wed jou hy gebruik 'n mikro om al die kinders se name te onthou!"

```
5 INPUT A
7 INPUT B
9 PRINT A + B
```

Gestel ons voer dié program uit. Na **reël 5** gaan die rekenaar vir 'n getal wag. As jy hom 'n getal gee, gaan dit daardie getal by hokkie **A** in die geheue bêre. By die volgende stap (7) wag dit weer vir 'n getal. Hierdie getal word in hokkie **B** gebêre. By die laaste stap (9) sal dit die som van die twee getalle gee.
Gestel jy gee vir **A** die getal **8** en vir **B** die getal **12**. Die skerm sal só lyk:



Ek dink hierdie program is heeltemal duidelik genoeg en dat dit vir jou geen probleme sal besorg nie. Let op dat die rekenaar die vraagtekens vertoon om jou te wys dat dit vir 'n getal wag. Sommige mikro's gebruik ander simbole in plaas van die vraagteken.
Die volgende program bereken die **oppervlakte van enige sirkel** (moet dit nie gebruik om jou huiswerk mee te doen nie!). By die eerste reël wag dit vir jou om die sirkel se straal te gee. As jy die straal intik, sal dit vir jou die oppervlakte van die sirkel gee.

```
5 INPUT STRAAL
7 LET AREA = 22/7 * STRAAL * STRAAL
9 PRINT AREA
```

Jy kan die program oor en oor gebruik deur elke keer **RUN** te tik.
Die volgende program bereken die **omtrek van enige sirkel**.
Die program vra jou om die straal en gee daarna die omtrek.

```
5 INPUT STRAAL
7 LET OMTR = 44/7 * STRAAL
9 PRINT OMTR
```

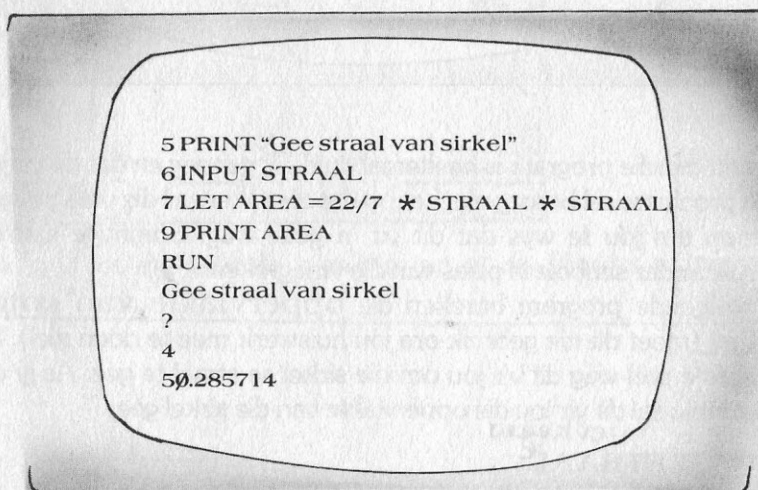
Daar is 'n ander slimmigheid van die **PRINT**-instruksie wat ons nog nie genoem het nie. Dit stel jou in staat om opskrifte of instruksies op die skerm te skryf. Om dit te gebruik plaas ons alles wat ons wil skryf **tussen aanhalingstekens**, byvoorbeeld:

```
7 PRINT "Dit was gister baie bewolk"
```

As jy nou die RUN-knoppie druk sal die volgende woorde op die skerm verskyn: "Dit was gister baie bewolk".
Jy kan dit saam met INPUT in 'n program gebruik. Neem weer die program wat 'n sirkel se oppervlakte bereken:

```
5 PRINT " Gee straal van sirkel"
6 INPUT STRAAL
7 LET AREA = 22/7 * STRAAL * STRAAL
9 PRINT AREA
```

As ons hierdie program uitvoer sal ons skerm só lyk:



Ons sien dat die sirkel se oppervlakte 50,2857.14 vierkante sentimeter is (as die straal 4 sentimeter is).
As jy nie wil hê dat almal jou programme moet gebruik nie, kan jy 'n geheime kode gebruik om dit te verseker. Plaas die volgende lyne vooraan enige program:

```
2 PRINT "Wat is die geheime kode?"
4 INPUT KODE
7 IF KODE <> 789 THEN STOP
1
1
1
```

die res van jou program volg hier.

In reël 7 gebruik ons iets wat julle nog nie voorheen gesien het nie. Die simbole <> beteken *ongelyk aan*. Jou geheime kode is 789. Indien iemand enige ander getal intik sal die rekenaar summier stop. As hy egter die korrekte kode intik, sal die program verder gaan.